GUIA PARA UN ANALISIS ESTRUCTURADO DE LA PROGRAMACION ESTRUCTURADA.

F. Sáez Vacas
Director de Formación de ERIA
Catedrático Numerario de Ordenadores Electrónicos de la E.T.S.I. Telecomunicación de Madrid.

GUIA PARA UN ANALISIS ESTRUCTURADO DE LA PROGRAMACION ESTRUCTURADA

Resumen: Se presenta un marco dinámico y contextual para la programación estructurada, válido para otras técnicas de informática. Si se quiere hacer síntesis el marco hay que considerarlo en su sentido "top-down", * pero si es de análisis de lo que se trata - y es de lo que se trata en esta ponencia - el sentido es "bottom-up" *.

1. INTRODUCCION

La programación estructurada es un tema de moda como lo fué, no hace mucho, el de memoria virtual. Tanto, que sospecho que pueda aplicársele, con las debidas modificaciones, la adivinanza que cita Teichroew (1) sustituyendo las palabras "decision tables" por "structured programming": ¿What do structured programming and algol have in common? y la respuesta, "both are more honored than used" que quiere decir, más o menos, que se habla mucho de ellos pero se utilizan poco.

Los actuales puntos de vista con respecto a la programación estructurada son tan confusos e incoherentes como corresponden a algo que está en su etapa de lanzamiento y por ello sometido a un régimen transitorio. Diferentes autores se encaraman en la onda del momento y nos ilustran, nos deleitan, nos orientan o nos

* Términos muy utilizados en la jerga de la programación estructurada.

desorientan con sus opiniones, experiencias y variaciones sobre el mismo tema. Tanto el estudioso como el usuario preocupado de mejorar su trabajo se muestran desorientados o en el otro extremo, excesiva y peligrosamente seguros en cuanto al contenido y significación de la programación estructurada. Los creadores de mitos echan leña al fuego, se marcan un tanto como especialistas al día y se preparan para agarrarse al próximo tema susceptible de mitificación. Lo que se echa de menos en todo este proceso es un análisis, con expresa definición del marco en que se sitúa el mismo.

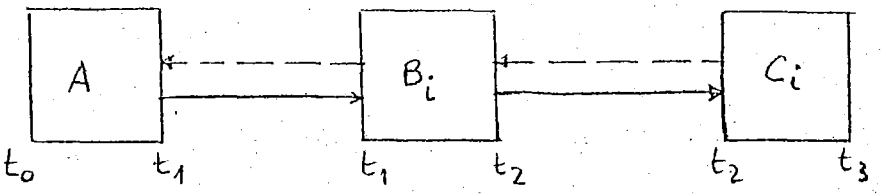
Muchos indicios señalan que el tiempo de respuesta del profesional ante un nuevo concepto o técnica de informática (no transparente) ha disminuído, si nos referimos a un plano verbal, pero posiblemente ha aumentado en el plano fáctico, o de uso real. Si esto es así, como supongo, la gran mayoría de los profesionales de la informática estarían asistiendo(asistiendo nada más) al gran espectáculo de la P.E. A mi entender, lo primero que se debe intentar, para que además de asistir se animen a participar, es situarles mínimamente en un marco adecuado de referencias. Todos sabemos que cuando se establece un marco para el estudio de un problema, automáticamente se aporta claridad y coherencia y de ello se induce una mayor rapidez en su resolución.

Aquí pretendo proponer una guía o modo de estructuración del análisis de la P.E., eventualmente de cualquier técnica informática, con el deseo de contribuir al acortamiento del tiempo de respuesta hacia su uso real.

2. EL LARGO VIAJE DEL FABRICANTE AL CONSUMIDOR

En plan abstracto y esquemático es lícito imaginar que una idea o una técnica se genera *, con ciertos grados de libertad, en un determinado contexto: laboratorio, centro de investigación, seminario ..., que llamaremos A. Durante una cierta cantidad de tiempo la idea es manipulada, desarrollada y perfeccionada en ese contexto. Llega un momento en que su nivel de desarrollo es suficiente para que sea captada por uno u otros contextos específicos de aplicación (Bi), en donde es sometida a nuevas pruebas, manipulaciones, especializaciones y complementaciones, al objeto de hacer de ella un elemento manejable por los consumidores finales. Estos ofrecen una resistencia a sustituir anteriores elementos por uno nuevo y transcurre otro intervalo de tiempo hasta ser aceptado, generalizado su uso, o rechazado provisional o definitivamente. El proceso que acabo de describir está representado (en el plano fáctico) en la figura 1, donde las flechas de trazo lleno indican el flujo de influencias y el sentido del tiempo. Por supuesto hay que admitir un efecto de "feed-back" o reacción (flechas de trazo) y también que las cosas no son tan simples, ya que hay solapamientos de los flujos. Pero desde un punto de vista global y en relación con mis intenciones, el esquema es suficientemente explicativo, porque resalta la variedad de contextos y el fluir del tiempo.

* A esto me refiero con la palabra "fabricante"



Según este razonamiento, aplicado en el mundo de la informática, habría una parte del personal activa en el contexto A, otra en el contexto Bi y otra, la mayoría, como consumidores de lo ideado por unos y transformado por otros, cada parte con su espacio de problemas, sus herramientas de trabajo y sus propios medios y formas de expresión. Creo que, en buena medida, el personal trabaja, habla, escribe, escucha o usa sin conocer su pertenencia funcional a un contexto determinado y sin conectarse debidamente con el flujo causa-efecto en el tiempo. El resultado es un barullo y su consecuencia más triste, la ralentización del flujo y, a veces, el agostamiento del mismo.

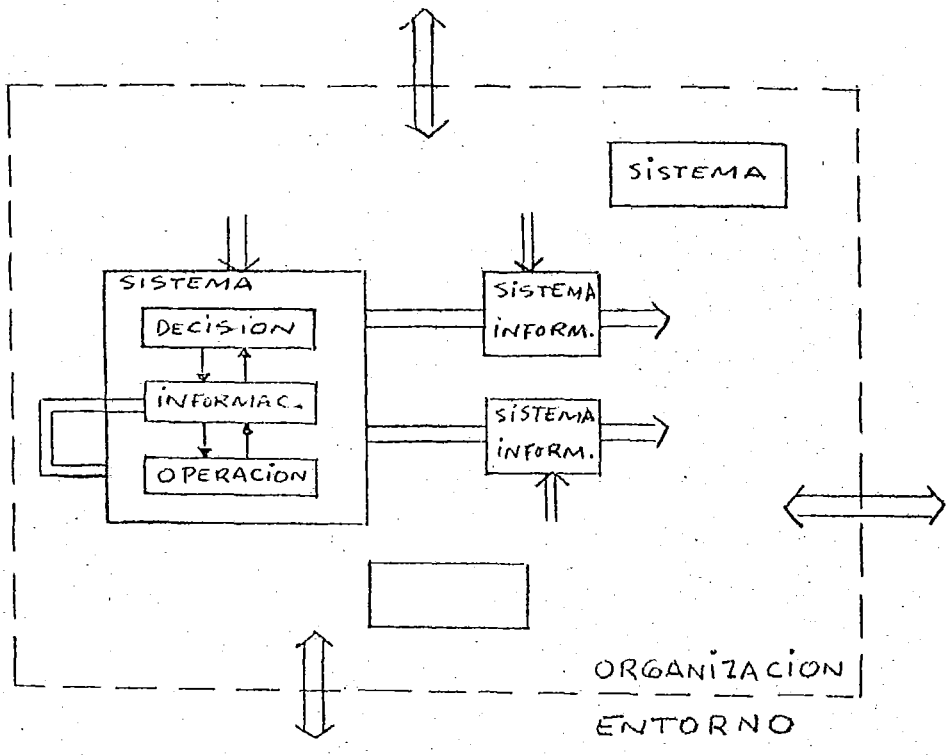
Tomemos el caso de la programación estructurada. La onda aún no se ha estabilizado en el consumidor, hablando naturalmente en términos estadísticos, y haciendo salvedad de la espectacular expectación que está despertando. Por consiguiente, la P.E., en su fluir,

ha llegado a C_1 y está en su período t_2 t_3, pero al mismo tiempo nuevas elaboraciones en B_i y en A alcanzarán en un futuro a C_i y a B_i y C_i respectivamente, y este es el marco de referencia. Solo queda dar nombres y apellidos a los contextos y decidir el momento y lugar en que nos situaremos en el flujo de las técnicas de programación. Al menos, como primera aproximación.

Consideraremos tres contextos posibles: el A, o laboratorio de ideas y técnicas avanzadas, el B_1 correspondiente al espacio de aplicaciones a las que puede aplicarse el concepto de ciclo de vida de un sistema y el B_2, espacio del resto de aplicaciones, de gran variedad y difícil sistematización. Dedicaré, casi con exclusividad, mi interés al contexto B_1.

3. EL CICLO DE VIDA DE LOS SISTEMAS Y LA PROGRAMACION ESTRUCTURADA

No cabe duda de que el concepto de ciclo de vida de un sistema de información es algo que posee una cierta solera. Esta clase de sistemas, cuyo ciclo ha sido descrito bajo diferentes formas por muchos autores (1) y en raras ocasiones prácticas desarrollado metódicamente, intercambia información con los sistemas de decisión y de operación de la organización que, a su vez, están sumergidos en un determinado entorno (2).



Una organización puede ser estudiada como un sistema que comporta una jerarquía de subsistemas, cada uno compuesto de tres subsistemas: de decisión, de operación, de información. Si se centra la atención en cualquier subistema, que llamaríamos sistema, su entorno es el conjunto de los sistemas que no son subsistemas del que consideramos (3), (4). Un sistema importante para nosotros es aquél (llámese Centro de Proceso de Datos, Departamento de Informática, etc. y esté donde esté situado en la jerarquía de la organización) que produce y opera en todo o en parte sistemas de información, incluyendo el suyo propio. Estos sistemas de información pueden quizá constituir a su vez un sistema. Todos los sistemas citados tienen su entorno con el que se comunican, por consiguiente son sistemas abiertos (esta es su principal y olvidada característica) y sujetos a las presiones y variaciones de aquél. Hacer frente a éstas supone incorporar los mecanismos de control necesarios, con objeto de mantener, al menos, (homeostasis) los objetivos del sistema, lo cual requiere una importación

de energía externa. Desgraciadamente, no pocos Departamentos de Informática y Centros de producción de software se dedican más que nada a mantener los objetivos de los sistemas de información ya creados, y escasamente a producir algunos nuevos o a mejorar los objetivos (calidad y performances) (Sáez(5)p.67) antiguos. Ocurre frecuentemente, además, que la importación de energía en el sistema es excesiva en comparación con la exportación de nuevos o mejores sistemas de información y la balanza se desequilibra, cuando por añadidura el precio de la energía aumenta. En suma, el Departamento de Informática que, como hemos dicho, es asimismo un sistema abierto, tampoco puede mantener sus objetivos relativos, a no ser que reorganice sus subsistemas y sus procesos. Esto sólo puede hacerse a través de una adecuada toma de decisiones, entre las que se contará la incorporación de la "tecnología" * necesaria para aumentar la productividad y la calidad de los mencionados procesos y de sus productos, los sistemas de información.

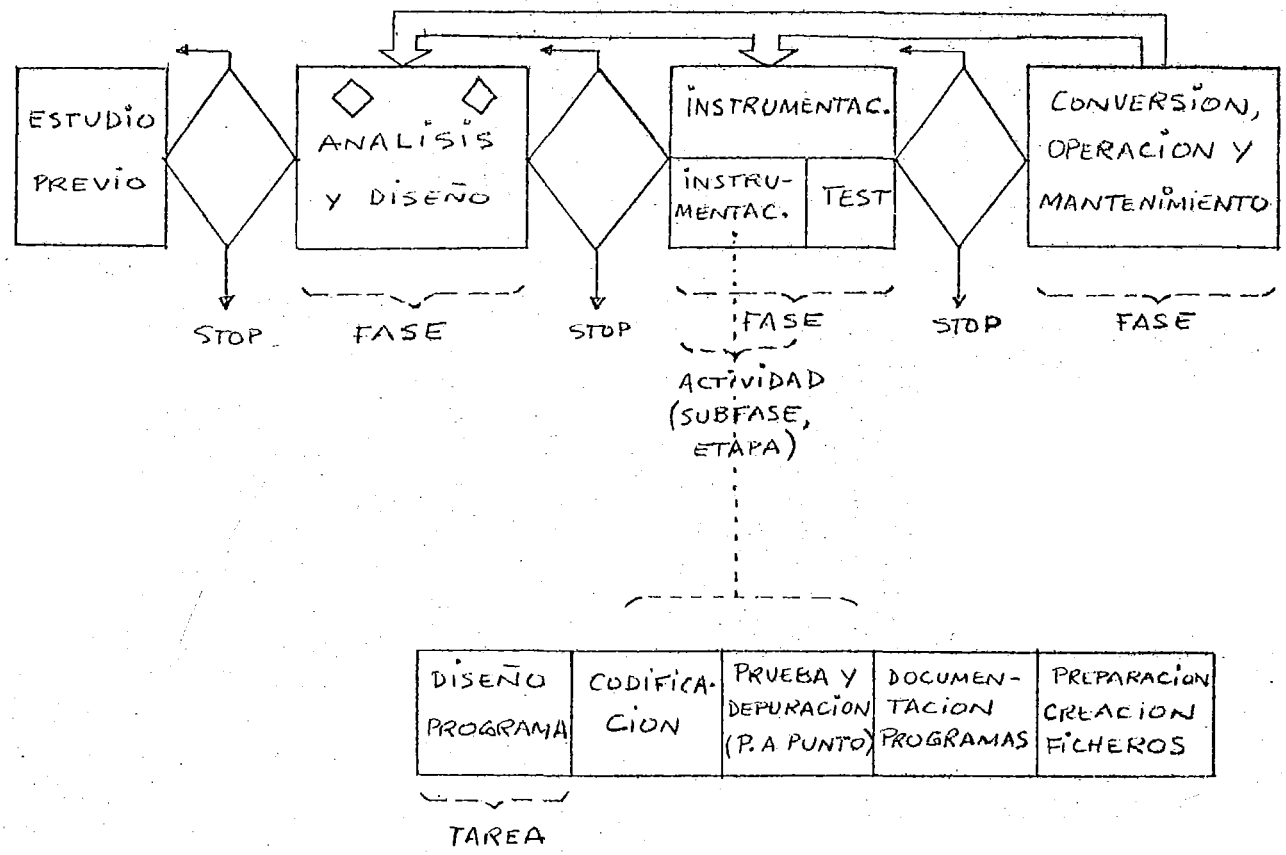
Pero vamos a los procesos, pues es en ellos donde encaja la P.E. vista ya como una pieza "tecnológica" con el suficiente grado de desarrollo para ser, por lo menos, analizada por el consumidor inserto en este contexto.

Siguiendo a Eriksen (6), el proyecto, construcción, operación y evolución de los sistemas de información es un proceso organizado

* En un sentido figurado. Especialemente no se refiere a los ordenadores.

de cooperación entre seres humanos y máquinas de tratamiento de la información que se desglosa en una secuencia de actividades, documentos y decisiones. Varias actividades constituyen una fase y las actividades pueden descomponerse en tareas (fig. 3). Otros autores, entre ellos Teichroew, proponen denominaciones y divisiones equivalentes.

CICLO DE VIDA DE UN SISTEMA



Realizar una tarea supone el empleo de unas técnicas, de manera que aquella está profundamente afectada por los cambios en éstas. En el contexto considerado, parece que una nueva técnica debe ser analizada desde el lado del consumidor, ente activo del ciclo, como un agente modificador de la tarea implicada, de la actividad, de la fase, de otras fases y hasta del conjunto del ciclo, cuya estructura puede quedar alterada hasta encontrar otro punto de equilibrio. Desde esta perspectiva relativista, la P.E. pide ser analizada, por extensiones sucesivas, al nivel de instrumentación (al que coge de lleno), al nivel de fase e interfases y finalmente al de ciclo, y analizarse, en cada caso, en relación con la etapa y los elementos correspondientes del proceso y con los productos resultantes de dicha etapa.

4. ACTIVIDAD DE INSTRUMENTACION DEL CICLO

Esta etapa, que también se puede llamar de implementación o de construcción, comprende las tareas de diseño de programas, codificación, puesta a punto, documentación de los programas, preparación y creación de ficheros y test del sistema. Es parte crucial y la única del proceso que se ve sometida a un control riguroso: el ordenador.

4.1. ¿Qué es la programación estructurada?. Sus principio o reglas fundamentales.

No existe, a mi conocimiento, ninguna definición rigurosa,

Gries (7), en una polémica con otro profesor universitario recoge hasta 14 distintas tomadas de la bibliografía sobre el tema, Wirth (8), probablemente uno de los más conspícuos científicos de la programación, emite la suya propia, no incluída entre las 14 anteriores: "Programación estructurada es la formulación de programas como estructuras jerárquicas de sentencias y objetos de computación". Hoy y aquí, nosotros (9) definimos qué es P.E. todo diseño de programas que respeta los siguientes principios:

1º. Estudia el problema por el procedimiento "top-down", es decir, de arriba abajo por niveles jerarquizados.

2º. Aplica unas estructuras básicas y predeterminadas

3º. Utiliza lo que se llama el recurso abstracto

Los dos primeros principios se han descrito con profusión en la literatura, no tanto el tercero que es tan vital como los anteriores *. Debido a ello, voy a recoger unos párrafos de Wirth (8). También puede leerse algo sobre este asunto en (17) en Dahl et al (23) y Dijkstra (40). Al recurso abstracto lo llamaba Djkstra en 1.969 (40) perlas que se ensartan sucesi-

* En particular, este principio puede tener repercusión trascendental en el diseño de hardware y software y especialmente en el camino hacia la programación automática.

vamente para formar un collar y en 1.972 máquinas virtuales o conceptuales.

Wirth (1.974): "Nuestra herramienta mental más importante para manejar lo complejo es la abstracción. Así, un problema complejo no deberá contemplarse inmediatamente en términos de instrucciones de ordenador, de bits y de "palabras lógicas", sino más bien en términos y entidades naturales al mismo problema, abstraidos en alguna forma adecuada. En este proceso se obtiene un programa abstracto que realiza operaciones especificas sobre datos abstractos y formulado en alguna notación apropiada, muy posiblemente en lenguaje natural. Se consideran entonces las operaciones como constituyentes del programa, que serán más adelante sometidos a descomposición en un próximo nivel más bajo de abstracción. Tal proceso de refinamiento continúa hasta alcanzar un nivel que pueda ser comprendido por un ordenador, ya sea un lenguaje de programación de alto nivel o bien un código de máquina.

Para el manejo intelectual es crucial que las operaciones constituyantes a cada nivel de abstracción sean conectadas de acuerdo con esquemas de programa suficientemente simples y bien comprendidos".

Los principios anteriores se pueden desarrollar o abordar de formas y con alcances diversos, lo que da lugar a distintas técnicas, métodos o modelos de programación estructurada.

4.2. Pequeña historia de las técnicas de programación estructurada en España.

Lo que sigue no es solamente una historia incompleta sino probablemente, subjetiva, dado el papel activo que he podido jugar en ella. En las tres últimas ediciones de Inforprim se ha tocado el tema de la programación estructurada. En el 1,973 se presenta una ponencia con este mismo título (10), quizás por primera vez utilizándose públicamente por estos pagos el mágico nombre de P.E., en donde se ofrecen los conceptos más importantes de la programación modular, de la programación estructurada en base a los tres esquemas de Dijkstra y del equipo estructurado de programación (chief programmer team). Al año siguiente, el atento consumidor que asiste a Inforprim recibe una somera descripción (11) de un modelo de programación estructurada con Cobol, modelo que, además, en opinión de sus autores (Bertini y Tallineau) asesta un severo palmetazo a una forma clásica de descripción de los programas: el organigrama (ordinograma, diagrama de flujo de tratamientos). (Esta ofensiva ha continuado fuera de nuestras fronteras con el apoyo de un libro (12) y un artículo (13), titulado espectacularmente con aires de esquela necrológica). En España la empresa consultora Delta Informática viene impulsando este modelo o uno muy semejante.

El autor de esta líneas, responsable y máximo entusiasta de la introducción en nuestro país, desde mediados de 1,970, de las Leyes de Construcción de Programas (LCP) de J.D. Warnier

y B.M. Flanagan((14), en fascículos de difusión interna en Honeywell Bull desde por lo menos 1.969, después traducidas al español, italiano, japonés, rumano, holandés e inglés), mientras lee los artículos del número de Datamation de Diciembre de 1.973 bajo el título genérico de "Revolutión in Programming" (15, 16,17 y 18) se pregunta si la ponencia(19) que sobre LCP presentaron miembros de su equipo en Inforprim 1.972 podría incluírse en éste movimiento revolucionario. La respuesta ha sido afirmativa, pero el punto de partida para el razonamiento propiciaba las dudas y las sospechas sobre sutilidades inaprensibles para mis colaboradores y para mí. Imagínense, un artículo de Datamation hablando de revolución, cuando la División de Educación de Honeywell Bull en España impartía cursos de programación estructurada a sus clientes (bajo el nombre de "metodología de programación") desde dos años antes (desde el 18 de Octubre de 1.971). En otro lugar de éste mismo volumen (20) se describen las relaciones del trabajo de Warnier con las estructuras de Dijkstra y con el modelo de Bertini, considerando decididamente a aquél como un modelo de programación estructurada, toda vez que respeta los principios enunciados más arriba (14, 19, 20), principios que hemos acuñado muy reciente y quizá provisionalmente (9).

Creo que lo más honesto es reconocer que el proceso para responder afirmativamente a la pregunta anterior ha sido largo y complejo. Lo primero fue localizar, obtener y estudiar al-

gunas publicaciones aparentemente significativas,por este orden (21), (22) y (23). Saltaba a la vista que estas últimas referencias y los trabajos de Warnier correspondían a dos mundos distintos, pero ¿ por qué, si ambos trataban de la estructuración de programas?.

Poco tiempo antes, la A.T.I. (Asociación de Técnicos de Informática) organizaba un curso (24) en el que, empleando sin vacilación el mombre de P.E., se discutían las LCP y la programación modular a través de un seminario (25) con HIPO, Cobol estructurado y LCP en el programa, si mis noticias son fidedignas. Mientras, la División de Educación de HB y todas las delegaciones de dicha compañía en nuestro país seguían impartiendo la LCP, bajo la denominación de Metodología de la Programación formando parte de un ciclo estándar de formación de programadores de gestión (26). Un organismo de formación de funcionarios públicos, la Escuela Nacional de Administración Pública, las introduce en su IV curso de Programación (1973-74), en plan de ensayo, y decididamente en el V curso (1.974-75), si no recuerdo mal con el nombre de Lógica de la Programación.

Fuera de nuestras fronteras, un artículo de Maleval (27) considera las LCP como un modelo de programación estructurada, pero EDP Analyzer (28) ("aunque existen algunos puntos de semejanza con la P.E., también parecen darse algunas importantes diferencias") y el propio Warnier no está de acuerdo, y éste último especifica en (29) dichas "importantes diferencias". Personalmente ya he dado mi respuesta y después haré

unas puntualizaciones al respecto.

Otros enfoques, para mi hasta ahora desconocidos y que, por consiguiente, no puedo describir ni enjuiciar son los que, impulsados por IBM España, utilizan las estructuras de datos y bloques de jackson y más recientemente la teoría de autómatas (ver (30)en este mismo volumen).

4.3. Varios modelos y ¿una sola y verdadera programación estructurada?

Aquí se puede hablar del bonito juego de la percepción humana, de la adaptación a un contexto y del tiempo, para explicarse los orígenes, los objetivos, las herramientas, las controversias en torno a la P.E. y la evolución de sus distintas apariencias. Hay un camino interpretativo que pasa por la psicología.

El fenómeno de la percepción es manejado con soltura por los psicólogos y por los expertos en comunicaciones humanas, pero muy poco o nada por los profesionales de otras ramas. Y sin embargo, recurrir a algunos conceptos muy simples y desprovistos de todo aparato especializado sobre la percepción y el pensamiento constituye una palanca para mover un mundo de ambigüedades en muchos casos. A fin de cuenta son seres humanos quienes inventan,adaptan o aplican las técnicas, luego en ello tiene que haber mucho de psicológico. Si utiliza-

mos alguna de las ideas emitidas por Fourastié, y recopiladas numeradamente en un anexo a esta ponencia (Ver anexo y Fourastié (31)), no tendremos ningún reparo en admitir que distintos autores hayan llegado con vehículos diferentes a lugares diferentes habitados por gentes diferentes, para ofrecerles cosas diferentes, respetando o coincidiendo en los mismos principios de la P.E. (F*1, F2, F3, F4):

Dijkstra. Formación en física teórica y matemática (32). Contextos A y B_2. Ambiente: universitario en investigación y enseñanza. Herramientas: razonamiento matemático. Experiencia en programación científica, sistemas operativos, Ciencia de la Programación. Más conocido a partir de mediados los años 60 y universalmente a partir de los primeros 70.

Warnier. Formación en humanidades. Contexto B_1. Ambiente: empresa constructora. Herramientas: Algebra de Boole y Teoría de conjuntos. Experiencia en fabricación de hardware, programación de máquinas tabuladoras (paneles con lógica cableada), aplicaciones de proceso de datos de gestión. Más conocido a partir de los primeros 70.

(*) La letra F seguida de un número indica la cita de Fourastié que figura en anexo y que apoya la idea que intento transmitir.

Bertini. Formación (?). Contexto B_1 y B_2. Ambiente: instituto de investigación relacionado con empresa constructora. Herramienta: lingüística, lenguajes. Experiencia: al menos últimamente, formación (11). Posterior a los dos autores citados.

Otros: Mismo procedimiento.

La circunstancia de cada autor condiciona su percepción de la realidad de la programación y sus métodos de ataque, así como su grado de proximidad en el tiempo y el grado de comunicación que puede esperar tener con el consumidor de un cierto contexto (F1, F2, F3, F4 y F6). Ahí es donde radican las "importantes diferencias" que se mencionaban en el punto 4.2. Las ponencias de Reyero y Rodero (20) y Martín (30) presentan distintos métodos de P.E. elaborados en distintos momentos del flujo de la P.E. y dirigidos a usuarios de C1. Estos tienen la elección.

Dijkstra queda, a mi modo de ver, como el principal y más elegante exegeta de los principios fundamentales de la programación estructurada (23), (32), (33), razón por la que nos apoyaremos frecuentemente en sus publicaciones.

Supuesto aceptada a estas alturas por el lector contexto C1) la existencia de unos principios universales de P.E. y

de modelos distintos de P.E. si quiere estudiar algunos de éstos se bifurcará así: Modelo Warnier, (14) o en plan más condensado (34), (19) y (20); Modelo Bertini, (12) y (13); Modelo Jackson-Teoría de autómatas, (30) y las fuentes recogidas en (30). Al analizar estos u otros modelos quizá puedan ayudarle, dependiendo de su intervención en el proceso de desarrollo de sistemas de información, las consideraciones que siguen.

4.4 Implicaciones de un modelo de P.E. en la estructura, eficacia y costes del proceso de instrumentación del sistema.

4.4.1 El diseño de programas, ¿arte o ciencia?.

Knuth (35), premio Turing 1974, nos dice que por ahora es tanto una ciencia como un arte y ambos aspectos se complementan bien.

Si ciencia significa "conocimiento ordenado lógicamente y sistematizado en forma de "leyes generales" y, por consiguiente, el enfoque científico viene caracterizado por palabras tales como lógico, sistemático, impersonal, racional, mientras que enfoque artístico se puede adjetivar mejor con palabras como estético, creativo, humano, intuitivo, no cabe duda de que el primer aspecto encaja más con la naturaleza industrial y económica del ciclo de vida de un sistema.

La vertiente creativa de la programación está perdiendo aparentemente batallas desde la aparición de lenguajes de alto nivel, pasando por los sistemas operativos y la memoria virtual, hasta llegar a los métodos de programación estructurada. El próximo paso, fuertemente impulsado por la P.E., será la programación automática. Pero lo que está ocurriendo en realidad es que las tareas más mecanizables de la programación (rutinas en el sentido cibernético de la palabra) se están mecanizando, sistematizando y, por último automatizando. Es un movimiento que sitúa la creatividad en capas

cada vez más altas del diseño, más cerca del problema que de la máquina. Por supuesto los problemas más sencillos en su estructura son los primeros en ser sistematizados (recordar texto de Wirth en punto 4.1).

Largos años de predominio de los intereses creados por la máquina (F7), han impreso carácter en muchos, que confunden creatividad con habilidad y les costará trabajo aceptar sustituir trucología por metodología.

Opino como Knuth, la programación es arte y es ciencia pero el problema aquí es cuál será el tiempo de respuesta (ver Introducción) necesario para vencer la barrera de muchos hábitos "artísticos" (personales, psicológicos) y sustituirlos en C_1 por métodos científicos (F2, F3, F4, F6). "Nadie sobrevive a la primera educación recibida sin algunas cicatrices. Según mi experiencia, la influencia intelectualmente degradante de algunos procesos educativos es una seria barrera para aclarar el pensamiento. A mis aspirantes les exijo, y no es broma, que no tengan ni idea de Fortran".- Dijkstra (36, pág. 22).

Aconsejo estudiar las formas o procedimientos que cada método propone para realizar esta tarea (organigrama, no organigrama (árbol, pseudocódigo), etc.) y contrastar con los hábitos o métodos en práctica.

4.4.2 La codificación.

En relación con esta tarea se plantean mayores fricciones técnicas. ¿Son los lenguajes actuales una barrera a la descripción de los programas estructurados (36)?. Como ocurre fatalmente se han elaborado antes los lenguajes que las técnicas de diseño de programas (F7, F8). El método de Warnier no establece ninguna limitación en el uso de los lenguajes *, mientras que Bertini ataca decididamente el problema de la adaptación del Cobol a la programación estructurada. En un ámbito más general se empieza a considerar peligroso el uso de la sentencia GOTO (22) y se acaba por crear lenguajes especialmente estructurados, como Pascal.

No pocas personas identifican la P.E. con una programación sin Goto, lo que, como hemos visto, no es cierto. Las dificultades más trascendentales que combate la P. E. son la programación de sistemas complejos y la falta de fiabilidad de los programas. Si solamente se tratase de eliminar el goto, no se resolvería más que una - parte del problema de la fiabilidad (ver Boehm (37) página 53). Además, desplazar lenguajes no estructurados pero muy extendidos (Cobol (59%), Ensamblador (20%), - RPG (6%), Fortran (5%), PL/1 (4%), otros (resto %) - - (38) parece ingente labor. Quiere decirse que, a corto plazo, cualquier método que imponga severas restricciones al uso de un lenguaje muy extendido encontrará resistencia.

(*) No olvidar la influencia de las características de los lenguajes en la productividad del programador y en la eficacia del código de máquina.

Los lenguajes estructurados que puedan diseñarse ahora tienen por delante un largo viaje hasta el consumidor (Ver punto 2), pasando primero por vehículo de comunicación más que de producción de programas (39) como le ha ocurrido, aunque en circunstancias bien distintas, al Algol (8).

4.4.3 La puesta a punto de programas y el test del sistema.

La falta de fiabilidad del software es un problema de enormes proporciones que han sentido en sus carnes todos los que utilizan un sistema operativo y los usuarios en relación con un software de aplicación. La prueba de un programa a través de un juego de ensayo no demuestra nunca la ausencia de errores, pues las condiciones de entrada hacen recorrer a la máquina solamente un número muy reducido de los posibles estados descritos por el código.

El conjunto de estados crece con la complejidad del programa. El programador, por su parte, tiene en su cabeza la posibilidad de multiplicar por un factor muy variable la dimensión del conjunto en función de su concepción del algoritmo. Esto en lo que concierne a un programa. Tratándose de un sistema completo, el número de combinaciones sube. Como expone Dijkstra (40, pág. 84) si un programa es un compuesto de N "componentes de programa", el nivel de confianza de los componentes individuales ha de ser excepcionalmente alto si N es muy

grande. Si los componentes pueden construirse con una probabilidad "p" de que sean correctos, la probabilidad de que el programa completo sea correcto no será mayor que $P = p^N$. Para un valor grande de N, p deberá ser prácticamente la unidad si se quiere que P no sea muy diferente de cero.

La P.E. reduce el número de combinaciones y en consecuencia tiene que reducir el número de errores; ¿en qué porcentaje?. Esto dependerá del método seguido y de otros factores. Existe poca perspectiva al respecto aunque se ha dado alguna experiencia. En casos sencillos se puede llegar a demostrar formalmente la corrección del programa, aspecto que es de los más investigados en la actualidad.

A título de ejemplo, el coste relativo de los trabajos de chequeo y test de un software complejo (arrastrando además todos los problemas de infiabilidad mencionados) supone aproximadamente un 40% del total de los trabajos de desarrollo del mismo (41) (37).

La productividad en estas tareas es intolerablemente baja a base de técnicas de programación no estructurada El lector se preguntará en qué medida cada uno de los métodos de P.E. puede influir en los factores considerados. Personalmente, conozco algunos resultados estadísticos en relación con el uso de las LCP dadas en (42) y recogidos en parte en este mismo volumen (20).

4.4.4 La documentación de programas.

Recordemos que el proceso reseñado en el punto 3 es una secuencia de actividades, documentos y decisiones. Los documentos aparecen como eslabón explícito en el proceso y su influencia es notoria a lo largo del mismo, aunque más bien se deja sentir en etapas posteriores a aquéllas en que ha habido que crear la documentación. Una buena documentación es siempre rentable, aunque es difícil que el autor de la misma, normalmente acosado por el corto plazo, lo perciba así y una mala o ninguna documentación implica en todos los casos consecuencias graves.

Desde este punto de vista es evidente la necesidad de enjuiciar el grado de comunicabilidad y las características documentales de cada método. Es un punto importante y configura uno de los factores de calidad de la metodología de trabajo. El programador no realiza un trabajo aislado, aunque lo realice solo, sino en conexión con los analistas, con otros programadores, con el personal de explotación, con los usuarios del sistema, consigo mismo. La documentación creada en un Departamento de Informática se integra en su propio subsistema de información que, recordémoslo (Punto 3), sirve de base a los subsistemas de decisión y de operación del mismo departamento. De ahí la importancia de que el subsistema de decisión comprenda la importancia de tomar la decisión de contar con un buen subsistema de información.

4.4.5 El humilde programador.

Nadie ha escrito posiblemente cosas tan bellas acerca de la figura del programador como Dijkstra (32) y Ershow (43). Ambos consideran la programación como un desafío intelectual de altura y a los programadores como un grupo de élite, "el primer grupo humano cuyo trabajo le lleva a aquellos límites del conocimiento humano marcados por problemas algorítmicamente insolubles y que tocan aspectos secretos del cerebro humano".

Aún más, Ershow concibe la programación como una actividad que "comprende ricos, profundos y nuevos principios estéticos sobre los que se basa la relación íntima de un programador con su profesión, y que le dan una satisfacción tanto intelectual como emocional.".

Estas reflexiones no me parecen poder aplicarse a la realidad de los contextos B_1 y C_1, por lo menos a la mayor parte de las actividades de programación que en ellos se realizan.

Las creo más adecuadas a otros contextos, aunque su carga es más bien normativa y describen lo que pudiéramos llamar las cotas más altas de la profesión.

Lo cierto es, sin llegar tan lejos, que la actividad del programador, piedra angular de la informática, no está debidamente encauzada ni valorada. La cuestión, desde esta ponencia, es sugerir interrogarse acerca de los efectos de los métodos de P.E. sobre el trabajo del programador actual y su curva de aprendizaje

(formación y adaptación al nuevo estilo) en función - de sus características aptitudinales y personales.

Otro aspecto interesante es el impacto que las nuevas técnicas de programación producen en esta actividad. La programación se empieza a parecer cada día más a - una ciencia y existe un amplio espectro de aportacio— nes para renovar este campo en cualquiera de los con— textos. Se avanza deprisa en la lógica de la programa- ción y en la psicología de la misma (44), desmenuzando la estructura de los procesos, la estructura de la con cepción de los procesos y hasta la estructura de las - condiciones óptimas (individuales y de grupo) para rea lizar los procesos. El concepto del "chief programmer team", que muchos han identificado con la P.E., es una organización en la que se utilizan de manera más ade— cuada las experiencias, saberes y aptitudes de varios individuos para constituir un sistema (decisión, opera ción, información) con el que resolver un proyecto im- portante. La herramienta de trabajo es la P.E., porque es necesario conseguir una comunicabilidad, una fiabi- lidad y una productividad mayores, pero el concepto es típico de "management" (por lo cual, no es extraño que pueda haberse aplicado en U.S.A. (18), (45)). Estos y otros estudios y experiencias sugieren jerarquías de - programadores, interesantes carreras y especializacio- nes en una actividad muy rica. La programación empieza

a liberarse de la máquina. Pero, en duro contraste, la situación real funciona en sentido contrario y nos encontramos con el hecho de que el programador es una figura en tránsito, en tránsito hacia otro puesto, por ejemplo, analista. Ni educacionalmente, ni laboralmente, ni socialmente, está conformada la situación para dar salida a las posibilidades promovidas por las nuevas técnicas. Aquí cabe preguntarse cuánto tiempo será necesario para voltear esta situación (F2, F3, F4, F6).

4.5 Implicaciones de un modelo de P.E. en la eficacia del Sistema.

Me refiero al resultado de cada una de las tareas del proceso. En la fase de instrumentación son interesantes las implicaciones sobre la fiabilidad de los programas, ocupación de memoria y tiempo de ejecución. Hoy es el primer factor el más trascendental. Los dos restantes han de ser sopesados con cuidado en lo que concierne a ciertas aplicaciones pero en términos generales van siendo superados por el coste de la tecnología de hardware y software de base. El último, tiempo de ejecución, que ha sido origen de hermosos malabarismos en programación, puede ya ser mirado, en una gran mayoría de casos, bajo un prisma coste/eficacia integrando en el coste el tiempo de programación (coste creciente).

5. FASES DEL CICLO.

Un método de P.E. altera en alguna forma las actividades de otras fases y no sólo, aunque sea en mayor medida, la de instrumentación. Con brevedad me referiré a las de operación y mantenimiento y de análisis y diseño del sistema.

En cuanto a la operación del sistema ya he citado los aspectos de comunicabilidad (con los usuarios y con el servicio de Explotación), a los que habrá que añadir la consideración del grado de adaptación con el sistema operativo. El mantenimiento promovido por evolución de las condiciones en que tiene que operar un sistema es necesario, podríamos decir que forma parte de los datos del problema, al tratarse de un sistema abierto. Si hablamos de software de aplicaciones, el tiempo dedicado a mantenimiento llega a consumir hasta 2/3 del tiempo de análisis y programación, según los resultados de una encuesta recogidos en (5), los que parecen dar por buenas las observaciones emitidas al final del apartado 3 de esta ponencia. Aquí puede repetirse el último párrafo del punto 4.4.3, a los que se puede añadir el papel crucial desempeñado en este terreno concreto por la documentación.

A nivel de fase no hay que considerar a ésta como un ente aislado, sino en relación con la que la precede y la que la sigue, es decir, las interfases. ¿Se ocupan o resuelven las interfases los métodos propuestos? ¿Cuáles y cómo?. Ya se ha dicho (punto 4.3) que los distintos modelos resuelven el problema de forma y con

objetivos distintos. Por ejemplo, los modelos de Warnier y de — Jackson realizan la estructuración de datos; el de Bertini la codificación utilizando un recurso real (Cobol). Preguntarse sobre la comunicabilidad con las actividades del análisis y el diseño. Los métodos de P.E., ¿están siendo prolongados por otros métodos coherentes en la fase de análisis y diseño?

Y por último, a nivel de ciclo, hay que meditar sobre las características relevantes de cada modelo y sus implicaciones en la planificación, organización y coste del desarrollo y con la eficacia, fiabilidad, adaptabilidad, mantenibilidad y progreso del Sistema.

6. EL FUTURO DE LA PROGRAMACION ESTRUCTURADA.

No es cuestión de profetizar, pero sí de dejar constancia de que se ha desencadenado un movimiento, que se está trabajando en A y que lo que hoy es investigación llegará dentro de algún tiempo a los consumidores, quizá modificando los métodos que actualmente conocemos como modelos de P.E. u originando otros nuevos.

Nombres como Manna, Floyd, Mills, Wirth, Böhm, Nivat, no resultarán muy familiares a bastantes lectores, porque laboran en A o en B_i y se ocupan de problemas tales como semántica formal de lenguajes, demostración automática y verificación de pruebas, formalización de rasgos especiales de lenguajes, recuperación de errores en la compilación, etc. Buscan nuevos esquemas de programa o una nueva formulación de la programación de los ordenadores (46). Martínez Carrillo, en este mismo volumen (39), nos describe algunas de estas vías.

La A.C.M. (Association for Computing Machinery) de U.S.A. prevé y explícitamente lo declara, aunque sea en plan de ejemplo, que diversos grupos de trabajo en áreas de interés especial (S.I.A.) puedan ocuparse con distinta percepción y con distinto nivel de la programación estructurada (47, página 80). El "SIA" en Computer Science and Technology se ocuparía de promover y transmitir entre sus miembros los conceptos de P.E.; el "SIA" en Computer Management and Personnel de cómo aplicarlos; el "SIA" en Uses and Effects of the Computer de diseminarlos entre los programadores de aplicaciones; estas experiencias se combinarían con la

teoría original en cursos sobre P.E. desarrollados por el "SIA" en Computer Education.

Esperemos que ésto ocurra también entre nosotros, poniendo en marcha medios adecuados para llegar en un plazo razonable a los profesionales y conseguir un sustancioso acortamiento del viaje del fabricante al consumidor.

ANEXO. FOURASTIE dixit.

F1.- El rasgo más importante en el pensamiento del hombre es lo que llamo la unicidad de la idea clara, o sea, el hecho simple de que no puede prácticamente tener más que un solo pensamiento consciente y claro en un momento dado (p. 13) ... De ella se desprende la extremada dificultad para conocer la realidad del mundo, a pesar de la facultad de percibir algunos elementos por los sentidos.

F2.- El hombre hace lo que ha aprendido a hacer (103).

F3.- El stock de ideas anteriormente adquiridas prima sobre la experiencia.

F4.- Concibo seis etapas en la información del cerebro, a partir de la "sensación". 1. La selección de las informaciones no percibidas y de las informaciones percibidas. Esta selección es inconsciente, involuntaria. Resulta de la situación biológica: los cinco sentidos, las pulsiones instintivas del código genético, las informaciones anteriormente adquiridas (modelos o programas, ...). 2. La selección (eficaz o no) (voluntaria o no) de las informaciones percibidas rechazadas y de las informaciones percibidas aceptadas. Esta selección se hace por una evaluación, por un juicio sobre el interés de la información; interés que depende de las informaciones anteriormente almacenadas y del advenimiento de un vínculo de parentesco (o de una oposición) perceptibles entre la información almacenada y la nueva información (244 y 245).

F5.- La abstracción y la unicidad del pensamiento. Es un arma esencial al conocimiento, pero es solamente por nuestras deficiencias que tal arma es necesaria (102).

F6.- Para introducir en una cabeza una idea nueva de rango, n, es preciso o bien que esté de acuerdo con las líneas más generales (de rango (n - 1)) que dicha cabeza posee o bien introducir una nueva idea de rango n - 1 que esté de acuerdo con ella (116).

F7.- El hombre debe oponerse a la tendencia, que pone en él el pensamiento único a corto plazo, de hacerse una concepción técnica de los procedimientos técnicos, según la cual, no siendo las técnicas más que medios no tendrían ninguna influencia sobre los fines y sobre el comportamiento intelectual y moral del hombre (156).

F8.- Cada conjunto de ideas racionales, es decir, ligadas por articulaciones "si... entonces" es para el cerebro un programa de búsqueda de la información por los sentidos y por la memoria (236)... La trama del programa es necesaria, pero lejos de ser suficiente para la reconstitución de una realidad compleja, que no puede ser percibida más que analíticamente, sucesivamente (mientras que existe globalmente, simultáneamente, experimentalmente) (237) ... En otros términos, un programa racional es bueno o mejor que otro, en la medida que permite al cerebro percibir en lo real, o en la memoria, una mayor cantidad de informaciones útiles a la descripción, a la decisión y a la acción (242).

- REFERENCIAS -

(1) D. Teichroew
IMPROVEMENTS IN THE SYSTEM LIFE CYCLE.
IFIP, 1974.

(2) J.L. Le Moigne
LES SYSTEMES D'INFORMATION DANS LES ORGANISATIONS.
Presses Universitaires de France, 1973.

(3) J.J. Jacq; L. Jehanin
LA RENTABILITÉ DES SYSTEMES INFORMATIQUES DANS L'ENTREPRISE
Presses Universitaires de France, 1974. Capítulo 1.

(4) M. Palao
INFORMATICA DE GESTION PARA DIRECTIVOS.
Librería Técnica Bellisco, 1974. Capítulo 9.

(5) F. Sáez Vacas
MEMORIA DE CATEDRA. ORDENADORES ELECTRONICOS.
Sáez, Dic. 1973.

(6) S. Eriksen
STRUCTURE AND CONTENTS OF AN EDP PROJECT.
Nord-Data-69 Congress, Junio 1969.

(7) D. Gries
ON STRUCTURED PROGRAMMING. A REPLY TO SMOLIAR.
Communications of the ACM. Vol. 17, No 11, Nov. 1974.

(8) N. Wirth
FROM PROGRAMMING TECHNIQUES TO PROGRAMMING METHODS.
International Computing Symposium 1973
North-Holland Publ. Co., 1974.

(9) Acuerdo tomado en reunión privada con los Sres. E. Reyero; J. Martín Faba; S. Rodero.
Madrid, Abril 1975.

(10) J.M. Martínez de Ubago; F. Morales; J. Martín Faba
PROGRAMACION ESTRUCTURADA.
Inforprim, 1973.

(11) M.T. Bertini
LE COBOL STRUCTURÉ . UN MODELE DE PROGRAMMATION DE GESTION.
Inforprim, 1974.

(12) M.T. Bertini; Y. Tallineau
LE COBOL STRUCTURÉ: UN MODELE DE PROGRAMMATION.
Editions d'Informatique, 1974.

(13) M.T. Bertini; Y. Tallineau
L'ORGANIGRAMME EST MORT
Informatique et Gestion, No. 64, En-Feb. 1975.

(14) J.D. Warnier; B.M. Flanagan
ENTRAINEMENT A LA PROGRAMMATION. I. CONSTRUCTION DES PROGRAMMES.
II. EXPLOITATION DES DONNEES.
Les Editions d'Organisation, 1970.
Versión española: PROGRAMACION LOGICA.
E.T.A., 1973.

(15) D.D. McCracken
REVOLUTION IN PROGRAMMING. AN OVERVIEW.
Datamation, Dic. 1973.

(16) J.R. Donaldson
STRUCTURED PROGRAMMING
Datamation, Dic. 1973.

(17) E.F. Miller; G.E. Lindamood
STRUCTURED PROGRAMMING: TOP-DOWN APPROACH.
Datamation, Dic. 1973.

(18) F.T. Baker; H.D. Mills
CHIEF PROGRAMMER TEAMS
Datamation, Dic. 1973.

(19) L. Merino; P. Brox
METODOLOGIA DE LA PROGRAMACION.
Inforprim, 1972.

(20) E. Reyero; S. Rodero
LEYES PARA LA ESTRUCTURACION DE LOS PROGRAMAS Y LOS DATOS.
Inforprim, 1975.

(21) C. Böhm; G. Jacopini
FLOW DIAGRAMS, TURING MACHINES AND LANGUAGES WITH ONLY TWO FORMATION RULES.
Communications of the ACM, Vol. 9 No. 5, Mayo 1966.

(22) E.W. Dijkstra
GOTO STATEMENT CONSIDERED HARMFUL
Communications of the ACM, Vol. 11, No. 3, Marzo 1968.

(23) O.I. Dahl; E.W. Dijkstra; C.A.R. Hoare
NOTES ON STRUCTURED PROGRAMMING.
Academic Press, 1972.

(24) M. Barceló; M. Costa
CURSO DE PROGRAMACION ESTRUCTURADA
A.T.I. Barcelona, Mayo 1973.

(25) M. Costa; E. Pardo; M. Barceló
SEMINARIO SOBRE APLICACIONES PRACTICAS DE LA PROGRAMACION ESTRUCTURADA.
A.T.I. Barcelona, Nov. 1974.

(26) F. Sáez Vacas
CUSTOMER TRAINING ORIENTED TOWARD THE CUSTOMER IN THE SPANISH AFFILIATE. BASIS FOR AN EDUCATIONAL POLICY.
Train, Primer trimestre 1973. Honeywell Bull.

(27) J.J. Maleval
LA PROGRAMMATION STRUCTURÉE.
Informatique et Gestion, No. 59, Julio-Agosto 1974.

(28) EDP ANALYZER
IMPROVING THE SYSTEM BUILDING PROCESS.
EDP Analyzer, Vol. 12, No. 12, Diciembre 1974.

(29) J.D. Warnier
LOGIQUE INFORMATIQUE ET PROGRAMMATION STRUCTURÉE.
Comunicación privada, 31 Diciembre 1974.

(30) J. Martín Faba
TECNICAS DE DISEÑO DE PROGRAMAS MODULARES Y ESTRUCTURADOS
Inforprim, 1975

(31) J. Fourastié
COMMENT MON CERVEAU S'INFORME. INFORMATIQUE CÉRÉBRALE.
R. Laffont, 1974.

(32) E.W. Dijkstra
THE HUMBLE PROGRAMMER
C.A.C.M., Vol. 15, No. 10, Octubre 1972.

(33) E.W. Dijkstra
THE STRUCTURE OF THE "THE"-MULTIPROGRAMMING SYSTEM.
C.A.C.M., Vol. 11, No. 5, Mayo 1968.

(34) J.D. Warnier
LES PROCEDURES DE TRAITEMENT ET LEURS DONNÉES.
Les Éditions d'Organisation, 1973 .

(35) D.E. Knuth
COMPUTER PROGRAMMING AS AN ART.
C.A.C.M., Vol. 17, No. 12, Diciembre 1974.

(36) J.N. Buxton; B. Randell, editores
DISCUSION SOBRE SOFTWARE QUALITY IN "SOFTWARE ENGINEERING TECHNIQUES", POR E.W. DIJKSTRA.
Nato Science Committee, 1970.

(37) B.W. Boehm
SOFTWARE AND ITS IMPACT: A QUANTITATIVE ASSESSMENT.
Datamation, Mayo 1973.

(38) A.S. Philippakis
PROGRAMMING LANGUAGE USAGE
Datamation, Octubre 1973.

(39) J.A. Martínez Carrillo; N. Lahuerta
TENDENCIAS EN PROGRAMACION ESTRUCTURADA. ALGUNAS EXPERIENCIAS.
Inforprim, 1975.

(40) E.W. Dijkstra
STRUCTURED PROGRAMMING EN "SOFTWARE ENGINEERING TECHNIQUES".
Editado por Buxton y Randell
Nato Science Committee, 1970.

(41) R.W. Wolverton
THE COST OF DEVELOPING LARGE-SCALE SOFTWARE.
IEEE Transactions on Computers, Vol. C-23, No. 6, Junio 1974.

(42) P. Dumora
LOIS DE CONSTRUCTION DES PROGRAMMES. LE POINT DE VUE DES UTILISATEURS.
Informatique et Gestion, No. 52, Nov. 1973

(43) A.P. Ershov
AESTHETICS AND THE HUMAN FACTOR IN PROGRAMMING.
Datamation, Julio 1972.

(44) G.M. Weinberg
THE PSYCHOLOGY OF COMPUTER PROGRAMMING.
Van Nostrand, 1971.

(45) F.T. Baker
CHIEF PROGRAMMER TEAM MANAGEMENT OF PRODUCTION PROGRAMMING.
IBM Systems Journal, No. 1, 1972.

(46) H.D. Mills
THE NEW MATH OF COMPUTER PROGRAMMING.
C.A.C.M., Vol. 18, No. 1, Enero 1975.

(47) A.C.M.
RECOMMENDED FUTURE DIRECTIONS FOR ACM.
C.A.C.M., Vol. 18, No. 2, Febrero 1975.